Measuring and Privately Building Highly Predictive Blacklisting

Luca Melis
University College London
luca.melis.14@ucl.ac.uk

Apostolos Pyrgelis
University College London
apostolos.pyrgelis.14@ucl.ac.uk

Emiliano De Cristofaro
University College London
e.decristofaro@ucl.ac.uk

ABSTRACT

Collaborative approaches to network defense are increasingly used to predict attacks as well as to speed up their detection. For instance, with *highly predictive blacklisting*, one aims to forecast attack sources based on alerts contributed by *multiple* organizations. While collaboration helps discover groups of correlated attacks targeting similar victims, it also raises important privacy concerns.

To address this challenge, we introduce a novel privacy-friendly system whereby organizations are clustered together based on the similarity of their logs, without disclosing them in the clear. Entities in the same cluster only share relevant logs and build more accurate blacklists. At the same time, we investigate how to measure the effect of collaboration on prediction and find that the state-of-the-art (non privacy-preserving) system actually achieves lower accuracy than if organizations predicted based on local alerts only. Our experiments shed light on how to improve the quality of predictions by optimizing information shared across organizations, showing that our privacy-friendly methods markedly outperform non private tools both in terms of precision and recall.

1. INTRODUCTION

Filtering connections from/to hosts classified as suspicious or malicious is a defense practice commonly used to reduce the number and the impact of attacks. Due to the impossibility of performing expensive real-time computations on each connection, filtering is usually done via look-ups against periodically updated lists of suspicious hosts, aka *blacklists*. There are two basic approaches to building blacklists: one can either maintain a local list based on past observations, or periodically obtain lists formulated by large-scale alert repositories (e.g., DShield.org, myNetWatchman [1], DeepSight [38]) with the most prolific attack sources. Zhang et al. [40] denote these two approaches, respectively, as local and global worst offender lists (LWOL/GWOL), highlighting how the latter misses a lot of attacks as sources may choose their targets strategically, while the former is completely reactive. Consequently, they introduce the notion of *highly predictive blacklisting* whereby different entities send their logs to a central authority, which, in turn, returns customized blacklists based on relevance ranking, or, as in follow-up work, on implicit recommendation [37].

In general, collaborative approaches to threat mitigation, besides predictive blacklisting, are being increasingly advocated both in the public and the private sector. Efforts to promote information sharing are proliferating, e.g., with initiatives from the White House [39], CERT [6], RedSky Alliance [34], Wombat [11], and Facebook ThreatExchange [2]. However, information sharing prompts important privacy concerns, and organizations are often reluctant to participate, due not only to confidentiality reasons, but also trust, liability, and competitiveness concerns. Even sharing firewall logs could harm an organization's reputation (e.g., negligence in applying patches) or disclose sensitive information about customers or business practices [3, 8].

This motivates the need for privacy-friendly approaches to collaborative predictive blacklisting. We aim to enhance accuracy of predictive blacklists by letting organizations collaborate with each other, while minimizing the amount of information they disclose in the process. First, however, we take a closer look at existing non privacy-preserving tools [37, 40] and notice that prior work only focused on measuring the improvement in successfully predicted attacks (i.e., true positives), but failed to account for incorrect predictions (i.e., false positives and negatives). We find that the current state-of-the-art non-private system [37] actually yields a very large number of false positives, thus achieving very low precision (0.08 in our experiments). It also incurs a lot of false negatives, which makes the overall accuracy even lower than if organizations performed prediction based on local logs only ($F1 = 0.14$ vs 0.26). This highlights that access to more data does not necessarily imply better predictions, and we argue that we can improve accuracy while at the same time protecting privacy by optimizing information disclosed by collaborating organizations.

We introduce a scalable and privacy-friendly system which relies on a semi-trusted authority, or STA, acting as a coordinating entity to facilitate clustering without having access to the raw data. Specifically, the STA clusters contributors based on the similarity of their logs (without accessing these logs), and helps organizations in the same cluster to share relevant logs. Toward this goal, we investigate (i) how to cluster organizations, (ii) what should be shared among them, and (iii) how to measure the effect of collaboration on accuracy. We experiment with four clustering algorithms – agglomerative clustering, k-means, k-NN, and DBSCAN – relying on the number of common attacks as a measure of similarity, which is computed in a privacy-preserving way. Then, we experiment with privacy-friendly within-clusters sharing strategies: only disclosing the details of common attacks and/or privately discovering correlated attacks.

We present the result of a comprehensive measurement analysis based on alerts obtained from DShield.org, involving 70 organizations which report an average of 4,000 daily events over a 15-day time window – an experimental setting mirroring a reasonable real-world deployment of the techniques. (Note that experiments on other periods yield similar results). For each clustering algorithm considered, and for each log sharing strategy, we compare the performance in terms of precision, recall, and F1 measure, as well as the increase in true positives, false positives, and false negatives

as a result of collaboration (i.e., comparing to a local baseline prediction). We observe that different clustering algorithms exhibit different behaviors, e.g., fluctuations are observed in the average improvement with some organizations benefiting more than others. We demonstrate that controlled data sharing helps organizations forecast attacks, compared to performing predictions locally. Sharing common attacks, and using events from previously unseen attackers that have cluster-wide correlation, maximizes the number of attacks predicted but also introduces more false positives.

Our privacy-friendly techniques markedly outperform existing algorithms for predictive blacklisting [37] in both precision and recall, demonstrating that our approach not only protects privacy and minimizes information exposure but also results in better predictions. We obtain 0.84 recall, 0.23 precision, and 0.36 F1 measure, compared to, respectively, 0.66, 0.08, 0.14 from [37].

In summary, we make three main contributions: (1) we show that what was considered the state-of-the art (non-private) system for highly predictive blacklisting achieves very low accuracy (Section 4); (2) we investigate how to measure the effect of collaboration on prediction and propose a novel privacy-friendly system for collaborative predictive blacklisting which performs even better than the non-private one (Section 5); (3) we show how to optimize the overhead incurred by the privacy protection layer using scalable cryptographic protocols (Section 6).

2. RELATED WORK

2.1 Collaborative Intrusion Detection

Katti et al. [22] are among the first to measure *correlated* attacks – i.e., attacks mounted by the same sources against different networks – establishing that they are very common yet highly targeted. They show that attack correlation persists over time and suggest that real-time collaboration between victims could significantly improve malicious IP detection time. In [40], Zhang et al. introduce highly predictive blacklisting, having different organizations contribute alerts to a central repository such as DShield.org, which in turn provides them with daily personalized (predictive) blacklists. The prediction uses a relevance ranking scheme similar to PageRank, measuring the correlation of an attacker to a contributor based on their history as well as the attacker's recent log production patterns. Then, Soldo et al. [37] improve on [40] using an implicit recommendation system to discover similar victims as well as groups of correlated victims and attackers. In their model, the presence of attacks performed by the same source around the same time leads to stronger similarity among victims, and a neighborhood model (k-NN) is applied to clusters similar victims. Cross Association (CA) co-clustering [7] is then used to discover groups of correlated attackers and victims, and prediction within the cluster is done via a time-series algorithm – Exponentially Weighted Moving Average (EWMA) – capturing attacks' temporal trends.

Meng et al [28] present a comprehensive survey highlighting the essential components, and the challenges, of collaborative security. The survey analyzes three key factors of collaborative intrusion detection, namely, communication, robustness, and privacy, arguing besides efficient and scalable communication, mechanisms for robustness (i.e., resilience to insider attacks) and privacy should be carefully considered.

Felegyhazi et al. [15] perform proactive prediction of malicious domain use: starting from a seed (e.g., confirmed bad domains), they predict clusters of related domains based on name server features (zone files containing subdomains and authoritative name servers), and infer new bad domains. Then, Liu et al. [26], based on externally observable properties of an organization's network, aim to predict breaches without the organization's cooperation. The system collects features about the organization's network (e.g., misconfigured DNS or BGP entries within a network, spam, phishing, etc.) and uses these features to train a Random Forest classifier.

Finally, Sirivianos et al. [36] propose a collaborative system that enables hosts with no email classification functionality to check whether a host is a spammer or not. Each host then assesses the trustworthiness of spam reporters by auditing their reports and leveraging the social network of the reporters' administrators. Whereas, Moura et al. [29] evaluate how network administrators can leverage different "bad neighborhood blacklists" (i.e., malicious hosts concentrated in certain portions of the IP address space which often correspond to poorly managed networks) generated by third-party sources to detect spam messages, and observe a significantly large intersection between third-party blacklists.

2.2 Privacy In Collaborative Intrusion Detection

In [33], Porras and Shmatikov discuss privacy risks prompted by sharing security-related data and propose data anonymization and sanitization to address them. However, follow-up work [10, 25] demonstrates that these techniques make data less useful and anyway prone to de-anonymization. Burkhart et al. [5] introduce a few privacy-preserving protocols based on secure multiparty computation (MPC) for aggregation of network statistics.This is also explored in [4] where entities send encrypted data to a central repository that aggregates contributions. However, statistics only identify most prolific attack sources and yield global models, which, as discussed in [40], miss a significant number of attacks and yield poor prediction performance. Nagaraja et al. [30] propose an inference algorithm, BotGrep, geared to discover botnet hosts and links in network traffics by identifying structured topologies, and partially rely on privacy-preserving algorithms, such as Private Set Intersection [13].

Finally, Freudiger et al. [17] focus on privacy-friendly collaborative threat mitigation, but rather than building an actual system, they focus on identifying which metrics (e.g., number of common attacks, dataset similarity) could be used to privately estimate the benefits of collaboration between two organizations, showing that only sharing information about common attacks already helps.

3. PRELIMINARIES

3.1 Dataset

Aiming to design a meaningful empirical analysis of highly predictive blacklisting, we gather a dataset of blacklisted IP addresses from DShield.org, as done in previous work [17, 37, 40]. DShield is a collaborative firewall log correlation system to which various organizations volunteer daily alerts. Each entry in the logs consists of a pseudonymized *Contributor ID* (the target), the *source IP* address (the attacker), *source* and *target port* number, and the *timestamp*. An example of an entry log is illustrated in Table 1.

Contributor ID	Source IP	Source Port	Target Port	Timestamp
...D982918	104.217.230.059	6000	1433	2015-06-06 11:49:32

Table 1: Example of an entry in the DShield logs.

Crawling. With DShield's permission, we have been collecting daily logs using a JavaScript web crawler, gathering, on average, 10 million logs from 120,000 organizations every day. We exclude entries for invalid, non-routable, or unassigned IP addresses, and discard port numbers, then, for each IP address, we extract its /24

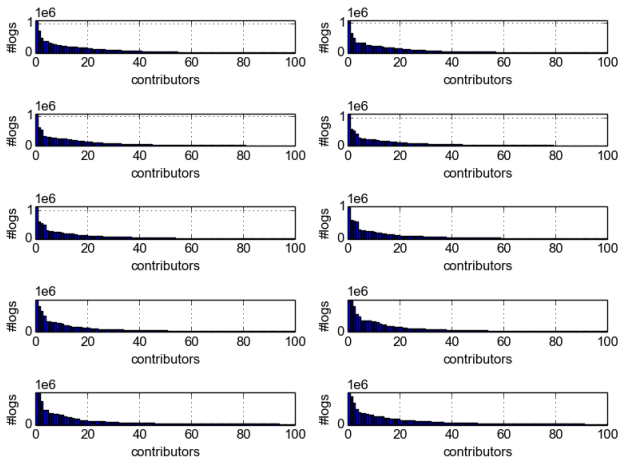


Figure 1: Number of logs during the 10 training-set windows of the period May 17–31.

subnet and use /24 addresses for all experiments. Note that this does not necessarily mean that predictive blacklisting algorithms will blacklist entire /24 subnets. This is an experimental choice also made by previous work [37, 40], as the main research goal is to compare the impact of different approaches on prediction. Moreover, blacklisting an address does not imply blocking all its traffic, but rather subject it to further scrutiny, e.g., enforcing rate limiting or only allowing outgoing packets.

Dataset. To facilitate our experiments, we select a 15-day period, May 17–31, 2015 and restrict our evaluations to a reasonably-sized sample of regularly contributing organizations. We select the top-100 contributors, based on the number of unique IPs reported, that also report logs every day during the 15 days. We then plot the number of logs contributed by each of the 100 organizations during each 6-day time window between May 17–31, 2015. As illustrated in Figure 1, most contributors (around 60) submit less than 100K logs, while fewer (around 20) submit between 100K and 500K, and only a handful of organizations contribute very large amounts of logs (above 1M). Then, we pick 70 organizations, for each time window, leaving out the top-10 and the bottom-20 contributors. We do so, like in previous work [37, 17], to minimize bias. Our final sample dataset includes 30 million attacks, contributed by 118 different organizations over 15 days, each reporting a daily average of 600 suspicious (unique) IPs and 4,000 attack events. ***Important Remark:*** Although this setting reasonably mirrors a plausible real-world deployment of our techniques, we have also repeated all our experiments on a larger number of organizations (150) and on two more sets of DShield logs, using, respectively, 15-day periods from February and December 2015, but have not found any significant difference in the results. Therefore, we limit the discussion of our experimental results to May 17–31.

Training and Testing Sets. We use the DShield dataset both as a *training* set and a *testing* set (i.e., ground truth). Specifically, we consider a sliding window of 5 days for training and 1 day for testing, as done in [37].

Notation. We use notation $\mathcal{O} = \{O_i\}_{i=1}^{n}$ to denote a group of n organizations, where each O_i holds a dataset D_i of alerts, i.e., suspicious IP addresses along with the related timestamp. We aim to predict IP addresses generating attacks to each O_i in the next day, using, as the training set, both its local dataset D_i, as well the set D_i', with suspicious IP addresses obtained by collaborating with other organizations. As discussed above, we consider $n = 70$ organizations using alerts collected from DShield.

3.2 EWMA Time Series Prediction

We use Exponentially Weighted Moving Average (EWMA) to perform prediction. Given a signal over time $r(t)$, we indicate with $\tilde{r}(t+1)$ the predicted value of $r(t+1)$, given past observations $r(t')$ at time $t' \leq t$. The predicted signal is computed as:

$$\tilde{r}(t+1) = \sum_{t'=1}^{t} \alpha \cdot (1-\alpha)^{t-t'} \cdot r(t') \quad (1)$$

where $\alpha \in (0,1)$ is a smoothing coefficient, $t' = 1, \ldots, t$ denotes the training window, and $t+1$ is the time slot to be predicted. For small values of α, EWMA aggregates past information uniformly across the training window, while, with a large α, the prediction algorithm focuses more on events taking place in the recent past.

3.3 Metrics

Throughout our evaluations, we use the following metrics to evaluate the performance of the predictions.

True and False Positives. For each time window and for each organization, we count *True Positives* (TP) as well as *False Positives* (FP). A TP occurs when the prediction algorithm includes an IP address in an organization's predictive blacklist that does appear in its testing set, and a FP – when it does not.

False Negatives. For each time window/organization, we generate predictive *whitelists*, i.e., sets of IPs that are not likely to attack an organization the next day, and count a *False Negative* (FN) when a whitelisted IP address instead appears in the testing set.

TP Improvement and FP/FN Increase. We also measure the average improvement/increase in TP, FP, and FN when compared to a baseline local approach, i.e., when no collaboration occurs between organizations and each of them makes its predictions based only on its local dataset. The improvement in TP is calculated as: $TP_{impr} = (TP_c - TP)/TP$ where TP_c is the number of true positives after collaboration and TP without. Similarly, the increase in FP and FN is denoted, resp., as $FP_{incr} = (FP_c - FP)/FP$ and $FN_{incr} = (FN_c - FN)/FN$.

Precision, Recall and F1-Score. We calculate the True Positive Rate (TPR), aka *recall*, False Positive Rate (FPR), as well as Positive Predictive Value (PPV), aka *precision*, defined as:

$$TPR = TP/(TP+FN)$$
$$FPR = FP/(FP+TN)$$
$$PPV = TP/(TP+FP)$$

and derive the F1 measure, i.e.,

$$F1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR}$$

Remarks on FP: The absence of an IP from our testing set can occur *either* when the IP is not considered suspicious *or* if it does not generate requests. While we cannot actually distinguish between the two cases, in the latter a FP is actually less "severe" than in the former, thus our FP count may be a bit more conservative. However, our main goal is really to measure and compare with each other the impact of *different* collaboration strategies on predictions so we use this method without loss of generality.

4. EVALUATING EXISTING PREDICTIVE BLACKLISTING APPROACHES

In this section, we present the results of an experimental evaluation of the implicit recommendation based approach proposed by

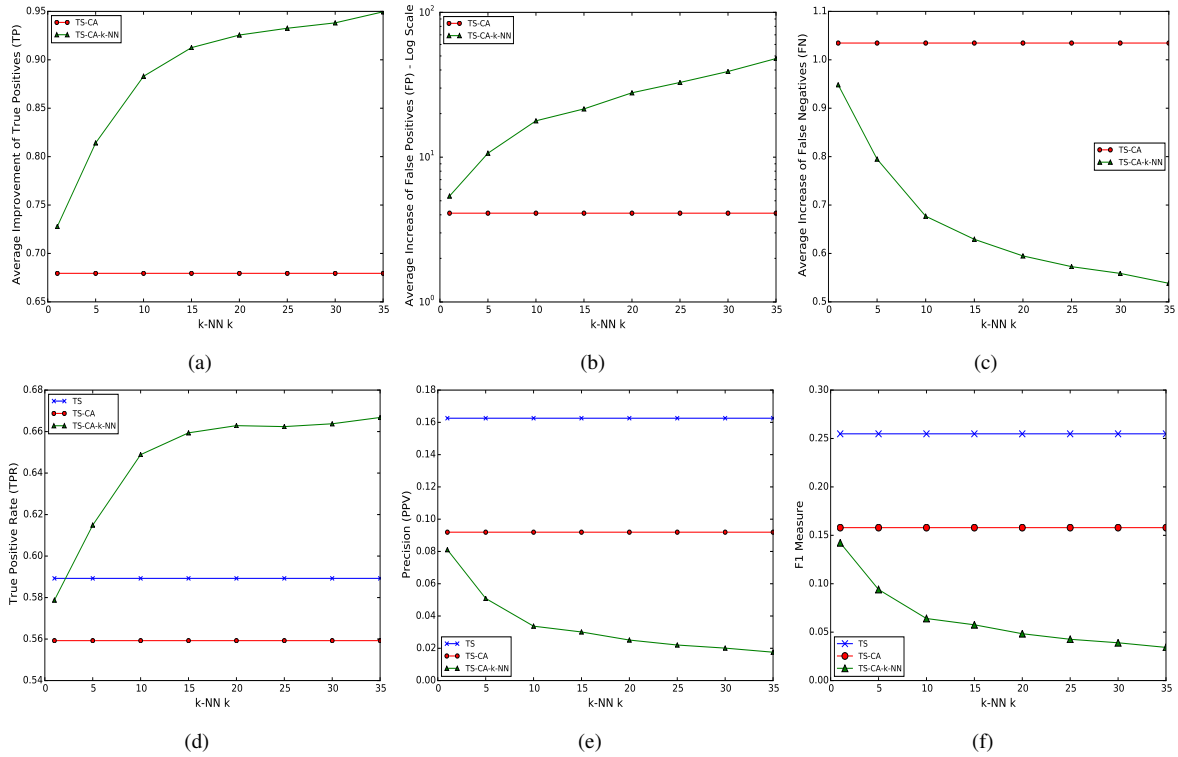


Figure 2: Soldo et al. [37]: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

Soldo et al [37], which is considered the state-of-the-art for highly predictive blacklisting. We have re-implemented their system in Python and used Chakrabarti's implementation of Cross Association (CA) as per [7].[1] More specifically, we evaluate [37]'s performance over our DShield dataset from Section 3.1: we start by measuring, as a baseline, the basic predictor which only relies on a local EWMA time series algorithm (TS), using $\alpha = 0.9$ as it yields the best results. Then, we apply their co-clustering techniques performing TS on clusters of correlated attackers and victims (TS-CA), and, finally, implement their full scheme by combining k-NN to cluster victims based on their similarity with CA and TS (TS-CA-k-NN).

Figure 2 illustrates the improvement/increase in TP, FP, FN (compared to the TS baseline) as well as TPR, PPV, and F1, with various k values (ranging from 1 to 35) used by the k-NN algorithm to discover similar organizations. Obviously the k-NN parameter k does not affect TS-CA and TS. Figure 2(a) shows that, with TS-CA-k-NN, the number of TP improve significantly as k increases, almost doubling the "hit count" compared to the TS baseline, whereas, TS-CA improves less (0.67). On the other hand, however, FP increase too, 5- to 50-fold, as clusters become bigger (Figure 2(b)), and naturally, this stark increase in FP leads to low precision, as shown in Figure 2(e). False negatives also always increase compared to TS (Figure 2(c)), specifically, they double with TS-CA and increase between 0.55 and 0.95 (less for larger k values) compared to TS. FN_{incr} also affects recall (Figure 2(d)), with an increase between 0.58 and 0.66.

Observe that the TP increase does not correspond to a comparable increase in TPR, due to the poor FN performance, as shown by the fact that TS-CA-k-NN reaches 0.95 in TP_{impr} but only at most 0.66 TPR compared to 0.59 with the baseline TS. Overall, Soldo et al.'s techniques achieve poor F1 measures, showing that their approach does not perform well in practice as TS-CA and TS-CA-k-NN yield lower F1 measures (at most 0.16 and 0.14, respectively) than a simple local time-series prediction (0.26).

[1] http://www.cs.cmu.edu/~deepay/software.html

5. PRIVACY-PRESERVING HIGHLY PREDICTIVE BLACKLISTING

5.1 Model

We set to support private computation of highly predictive blacklisting, so that organizations with limited mutual trust can collaborate and enhance the prediction of future attacks, while only the minimum amount of information is disclosed in the process. We assume the presence of a semi-trusted authority, namely STA, which acts as a coordinating entity between the organizations, but does not have access to data in the clear. In practice, STA could be run by DShield, or an ad-hoc cloud-based service, assisting organizations in the same sector (e.g., universities, banks, small businesses).

The STA performs clustering to identify organizations that are targeted by similar attackers, based on a pairwise similarity matrix, computed privately, which we denote as O2O (organization-to-organization). Once organizations have been assigned to clusters, they share information in a controlled way. Specifically, the system involves the following steps (detailed in Sections 5.1.1–5.1.4):

1. Organizations compute, *privately*, a pairwise similarity measure, used by STA to build an organization-to-organization matrix ("O2O");
2. STA clusters organizations based on O2O;
3. Organizations share, in a *privacy-preserving way*, data with other organizations in their cluster;
4. Organizations perform the prediction based on their "enhanced" datasets.

Our system follows a *data minimization* approach whereby organizations in the clusters only share information about common and correlated attacks. We assume that STA is a semi-trusted authority that does not collude with any organization to violate privacy of other parties. We also assume that all the parties are semi-honest and follow the protocol specifications without modifying their inputs (see Appendix A for more details).

5.1.1 Organization-to-Organization Similarity

In order to cluster together organizations that are targeted by similar attackers, we use a similarity measure to be computed between each pair of organizations O_i, O_j (on input D_i, D_j, respectively).

We consider a few approaches:

- The cardinality of the set intersection, $|D_i \cap D_j|$, i.e., the number of common attacks. Note that the intersection is between multi-sets, so if the same IP appears twice in both parties' logs in the same day, it is counted twice in the intersection;
- Jaccard similarity index $|D_i \cap D_j|/|D_i \cup D_j|$;
- Cosine similarity, assuming datasets have size m:
$$\frac{\sum_{k=1}^{m} d_{i_k} d_{j_k}}{\sqrt{\sum_{k=1}^{m} {d_{i_k}}^2} \cdot \sqrt{\sum_{k=1}^{m} {d_{j_k}}^2}}$$
- Pearson Correlation, assuming datasets have size m:
$$\frac{\sum_{k=1}^{m} (d_{i_k} - \bar{d}_i) \cdot (d_{j_k} - \bar{d}_j)}{\sqrt{\sum_{k=1}^{m} (d_{i_k} - \bar{d}_i)^2} \cdot \sqrt{\sum_{k=1}^{m} (d_{j_k} - \bar{d}_j)^2}}$$
where $\bar{d}_i$, $\bar{d}_j$ are the absolute means of D_i, D_j.

We have evaluated the four approaches above experimentally and have observed that using cardinality of set intersection yields the best results overall in terms of precision and recall. Especially, Pearson Correlation and cosine similarity do not produce good overall performances (in terms of TPR and PPV). Therefore, to ease presentation, we only present results using the cardinality of set intersection as a pairwise similarity measure between organizations. Note that, to support private preserving computation of set intersection cardinality, we can use Private Set Intersection Cardinality (PSI-CA) [12]. In Section 6, we will introduce a server-aided variant reducing complexity by minimizing public-key operations, thus providing full scalability.

5.1.2 Clustering

The next step is for the STA authority to perform clustering based on the O2O matrix. For this task, we consider four clustering algorithms – namely, agglomerative clustering, k-means, k-NN, and DBSCAN – which use different approaches to discover groups of similar organizations. As discussed above, entries in the O2O matrix are similarity measures between two organizations, defined as the number of IP addresses attacking them both over a given time window. To ease presentation, we defer the review of the clustering algorithms to Appendix B.

5.1.3 Log Sharing

Next, STA reports to each organization the identity of other organizations in the same cluster (if any), so that they collaboratively (yet privately) share logs to boost the accuracy of their prediction. Below, we consider a few possible strategies to share information.

Common Attacks (*Intersection*). Each organization in the cluster only shares logs corresponding to common attackers. Specifically, if an attacker is common for two organizations, these organizations share the events from their datasets related to this IP address. As a result, each entity enhances its local dataset with additional events about an attacker that have been witnessed by the rest of the contributors in its cluster, thus reinforcing the knowledge about that attacker. *Privately* sharing information about common attacks is possible via secure multiparty computation, using the Private Set Intersection with Data Transfer primitive (PSI-DT) [13](reviewed in Appendix A). Section 6 shows how to rely on a server-aided variant to achieve scalability in the presence of many organizations.

Correlated Attacks (*IP2IP*). We also consider letting organizations in the same cluster identify correlations between attackers, by building a matrix of co-occurrences of attack sources. We aim to capture correlations between attackers that cannot be discovered based on the individual logs of each contributor, or using the common attacks approach discussed above. More specifically, a symmetric matrix – which we denote as IP2IP matrix – is built to store the number of co-occurrences for each pair of attackers, and a clustering algorithm such as k-NN is run to find correlated attacks.

To perform this task in an efficient and *privacy-preserving way*, we rely on the protocols proposed by Melis et al. [27] – also reviewed in Appendix A – for privacy-preserving Item-KNN using Count-Min sketches [9]. Each organization in a cluster encrypts and transmits a *succinct* representation of their matrix in such a way that STA can only decrypt the aggregate matrix. Despite an upper-bounded error in the aggregate is introduced due to the succinct data representation, the communication and computational overheads incurred by the cryptographic operations are reduced from linear to *logarithmic* in the size of the IP2IP matrix. Once the STA obtains the aggregate IP2IP matrix, it runs the Item-KNN algorithm, finds correlated attacks, and communicate them to organizations in the same cluster.

Common & Correlated Attacks (*IP2IP+intersection*). Obviously the two strategies discussed above can also be combined, so that cluster contributors can benefit from sharing their intersection (i.e., obtaining more events on attackers that already exist in their training set) as well as obtaining events from previously *unseen* attackers that have cluster-wide correlation.

Baseline (*Local*). We will also consider a baseline approach whereby each organization does not share any information with any other entity, i.e., making predictions based only on its local logs.

Sharing Everything (*Global*). We also measure a global approach whereby all contributors in the cluster share everything with each other. While we expect this approach to provide the highest degree of intelligence to the organizations, some of the information may be irrelevant to some contributors, yielding a high number of false positives/negatives. As we use this approach simply for comparison, we do not take into account privacy protection.

5.1.4 Attacks Prediction

In order to model the temporal dynamics of the attacks, we use a time series approach, namely Exponentially Weighted Moving Average (EWMA, cf. Section 3.2), as [37] showed that, for the large majority of attacking IP addresses, future activity strongly depends on the recent past. EWMA predicts future values based on past values weighted with exponentially decreasing weights towards older values. Specifically, for each organization u and for each attacker a on its training set, $r_{a,u}(t')$ denotes a binary value indicating whether or not a attacked u at time t'. EWMA aggregates these signals from the training set and outputs a measure of how likely a is going to attack u again. When collaboration takes place, the training set of each organization is augmented with logs and events coming from the entities in its cluster. Using this "enhanced" training set, the organization trains the EWMA algorithm and creates augmented blacklists. Once again, we set $\alpha = 0.9$ as it yields the best results.

5.2 Comparing Strategies for Clustering and Log Sharing

We now present the results of our extensive experimental evaluation, comparing how different clustering (agglomerative, k-NN, k-means, DBSCAN) and sharing approaches (intersection, IP2IP, IP2IP+intersection) affect the performance of the predictive blacklisting, and compare to local (no sharing) and global (sharing everything) baselines. Our experiments are written in Python, using the scikit-learn machine learning suite [31].

Note: Our main goal is to measure the effect of collaboration on the prediction, seeking to improve TP while keeping the increase in FP at a low, even though – as we discuss in Section 3.3 – we count FP in a somewhat conservative way. We are interested in comparing different clustering/sharing approaches with each other, as well as to the state-of-the-art non-private system by Soldo et al. [37].

Settings. Throughout our experiments, we use notation and parameters introduced in Section 3, i.e., we consider 70 organizations using alerts collected from DShield in a 15-day period. Also, for the IP2IP method, we only consider the top-1000 attackers (i.e., the top-1000 *heavy hitters*) in each cluster, for each 5-day training-set window, rather than looking for correlations over all the /24 IP space. Note that this is only done to speed up our experiments, whereas, we can actually use the Count-Min sketch based private aggregation protocols from [27] to efficiently support k-NN computation, privately, over all the 2^{24} addresses. In fact, the succinct representation of the IP2IP matrix uses a sketch of size $O(\log(2^{24} \cdot 2^{24}))$: specifically, given parameters (ϵ, δ), we get a matrix of size $L = d \times w$ where $d = \lceil \ln(2^{24} \cdot 2^{24})/(2 * \delta) \rceil$ and $w = \lceil e/\epsilon \rceil$. Setting $\epsilon = \delta = 0.01$ [27], the Count-Min sketch size amounts to $L = 10,336$, yielding practical computational/communication overhead.

Finally, we fix the k value for the k-NN based recommendation to 50, as it provides the best results in our experiments.

5.2.1 Agglomerative Clustering

We consider different numbers of desired clusters (k), ranging from 1 to 35, setting affinity and linkage parameters to *cosine* and *average*, respectively, to indicate what distance measures to use between sets of observations. For $k > 15$, the size of the cluster slowly reaches 2 (see Appendix C for more details). As each organization is assigned to exactly one cluster, in each time window, all 70 organizations are involved in collaborations.

In Figures 3(a)–3(c), we plot average TP_{impr}, FP_{incr}, and FN_{incr}, respectively, with increasing number of clusters. Somewhat unsurprisingly, IP2IP achieves smaller TP_{impr} results (at most 0.20) than global (up to 1.10), which however incurs higher FN_{incr} (0.3) and above all FP_{incr} (1000). We also note that TP_{impr} presents a relatively large standard deviation for all cluster sizes, indicating that there are a few organizations that benefit from collaboration much more than others.

Figure 3(d) shows that recall (TPR) always improves when sharing, with intersection reaching 0.76. When combining intersection and IP2IP, TPR slowly degrades with smaller clusters (peaks at 0.68 for $k = 5$), while, with IP2IP, it increases (0.67 for $k = 35$). Because of the increase in FN, global performs worse in terms of recall (0.66) although obtaining the best TP_{impr}.

From Figure 3(e), observe that local yields the best performances in terms of precision (0.16), followed by intersection (0.15, slightly growing for larger k), while IP2IP and IP2IP+intersection slowly increase up to 0.13. Global performs poorly overall (up to 0.04) due to high FP. Finally, Figure 3(f) plots the F1 measure: intersection achieves slightly better scores than local (0.26 vs 0.25), while its combination with IP2IP, or just IP2IP are slightly worse (0.22).

5.2.2 k-means

Next, we use k-means (with O2O as the dataset of feature vectors) for clustering and obtain results similar to agglomerative clustering. Thus, we decide to restrict to *stronger correlations*, by only taking into account organizations closer to the cluster's centroid, and excluding the rest of them as outliers. We set a distance threshold and experiment with it empirically, finding that the optimal setting is the 40th percentile, i.e., the cluster distance value below which 40% of the organizations can be found. With growing values of k (ranging from 1 to 35), the algorithm produces more and smaller clusters, and more organizations collaborate (see App. C).

Figures 4(a)–4(c) plot the average improvement in TP and increase in FP and FN, respectively. TP_{impr} is somewhat constant with IP2IP (0.2) independent of the cluster sizes, while with the other methods it decreases faster due to the distance thresholds, ranging from 1.1 with global for $k = 1$ to 0.1 of intersection for $k = 35$. IP2IP shows steady FN_{incr} values compared to other methods (-0.2, i.e., a 20% decrease) which leads to a better performance in TPR, as shown in Figure 4(d), for $k \geq 10$ (up to 0.71). Furthermore, intersection yields the best performance in FN_{incr} (-0.52), with $k = 1$. Figure 4(f) shows the best F1 measure (0.30) is reached using intersection with $k = 5$, due to a peak both in PPV and TPR. IP2IP performs slightly worse (0.23) than local (0.28) while poor F1 values for global, with $k = 35$, (0.18) are due to its bad PPV (0.10) – see Figure 4(e).

5.2.3 k-NN

Recall that, in k-NN, the parameter k indicates the number of nearest neighbors that each entity considers as its most similar ones. Thus, organizations can end up in more than one neighborhood (unlike agglomerative clustering or k-means). Since the algorithm builds a neighborhood for each organization, not all clusters have the same *strength*. Therefore, we only consider *strong* clusters in terms of their members similarity and as done with k-means, after tuning the parameters, we set a distance threshold as the 40th percentile to leave possible outliers out of the clusters.

As k increases, so do cluster sizes, ranging from 1 to 14, and the algorithm builds on average 25 strong clusters in each time window. Note that for $k = 1$ all organizations are involved in exactly one cluster of size 1, since k-NN identifies them as their own nearest neighbor. Other than this special case, as we focus on strong clusters and set a distance threshold, the system makes fewer organizations collaborate overall (ranging from 220 with $k = 5$ to 320 with $k = 35$). See Appendix C for cluster sizes and the number of collaborators involved.

From Figure 5(a), we observe that IP2IP+intersection yields the second best performance in TP_{impr} (0.38, with $k = 35$), while global peaks at 0.60. Interestingly, the standard deviation of TP_{impr} is much smaller than with other clustering algorithms, indicating that, due to overlapping clusters, "big" contributors can help a lot "smaller" ones, as they are involved in multiple clusters, thus, improvements are more uniformly distributed over the contributors that are involved in collaboration. In terms of FP_{incr}, IP2IP doubles the number of FP (for $k = 35$), while intersection achieves the lowest value with 0.51 (again, for $k = 35$). As with previous clustering algorithms, we notice that intersection yields the best decrease in FN, i.e., -0.5 with $k = 35$.

Intersection also achieves the highest TPR (up to 0.77) with larger cluster sizes (i.e., for $k \geq 10$), while its combination with the IP2IP reduces it (0.71) – see Figure 5(d). Figure 5(e) shows that intersection has the best PPV too (0.19 for $k = 15$), similar to local

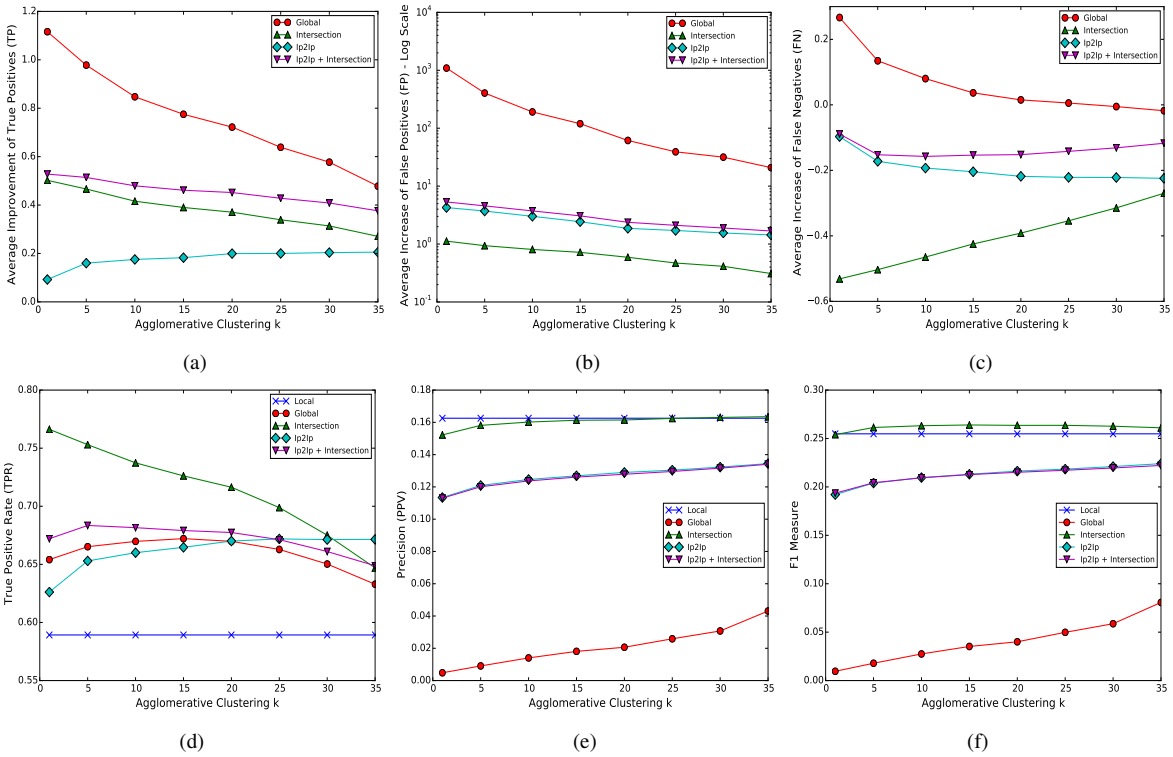


Figure 3: Agglomerative Clustering: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

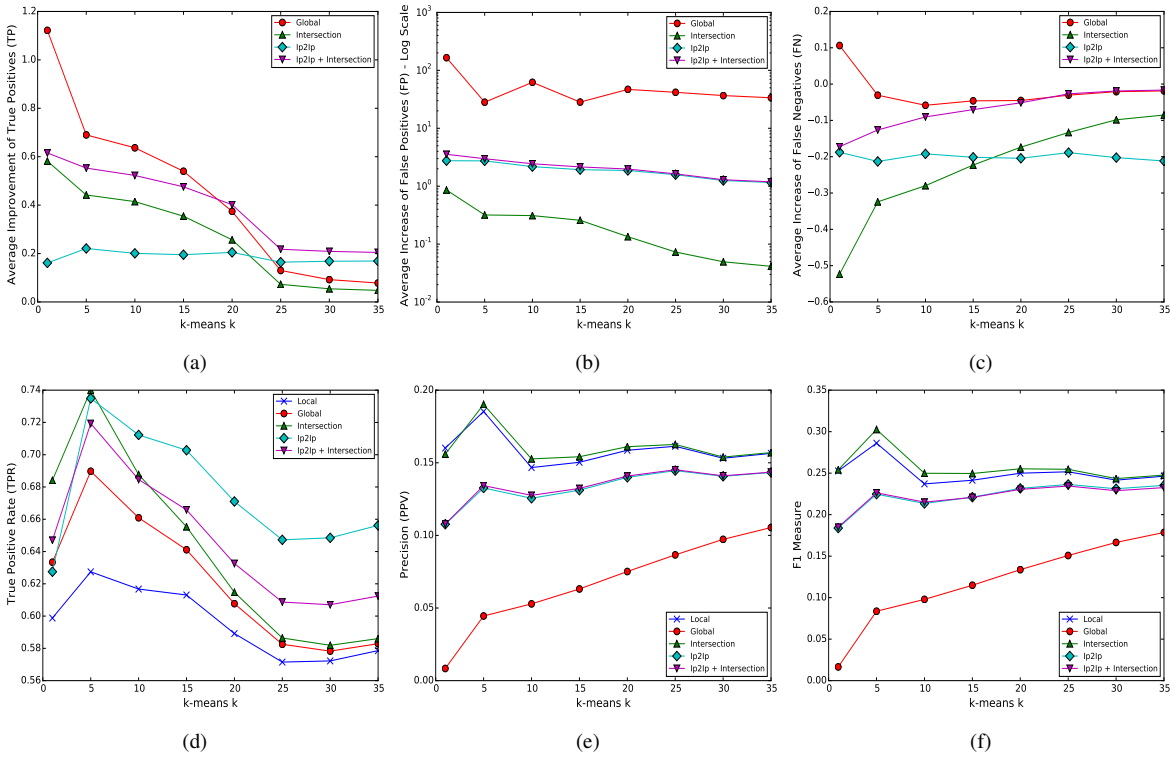


Figure 4: k-means: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

(0.18), while IP2IP performs worse (0.16) due to higher FP_{incr} (almost doubling the FP for $k = 35$). Finally, Figure 5(f) reports the F1 measures, with intersection obtaining the highest (0.30 for $k = 15$), and the other approaches yielding worse scores, i.e., 0.28 (local), 0.25 (IP2IP+intersection), and 0.13 (global).

5.2.4 DBSCAN

Our last set of experiments are with DBSCAN: unlike the other

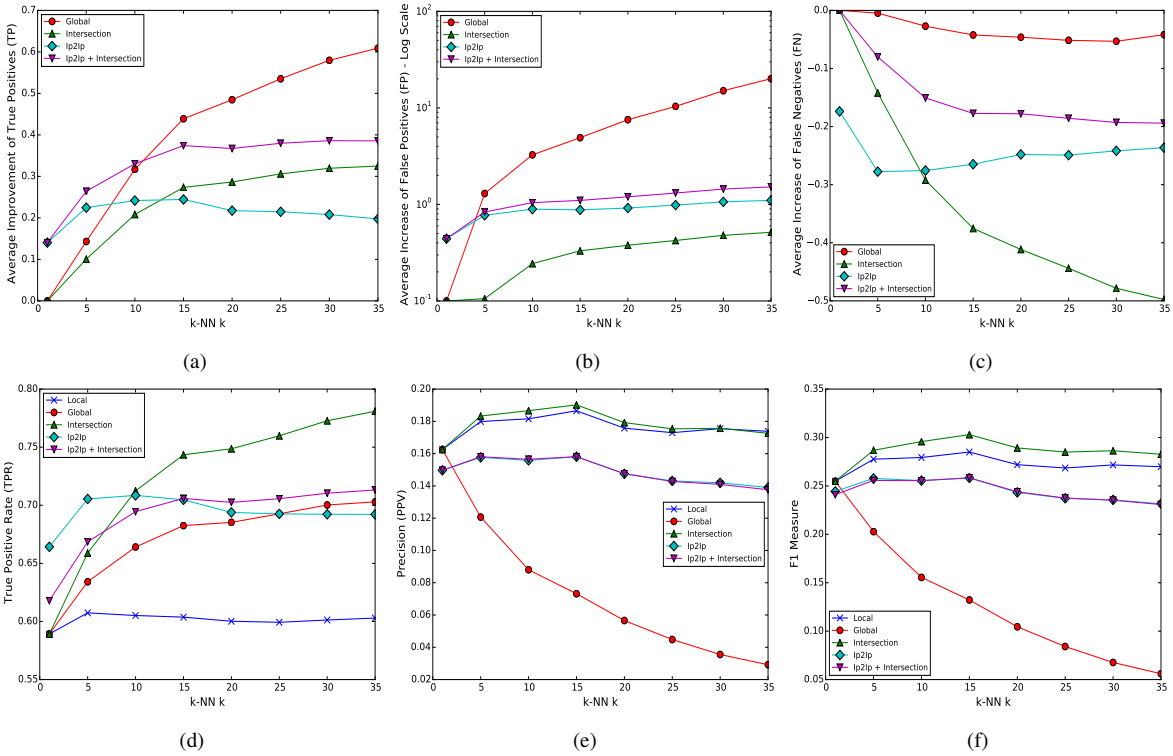


Figure 5: k-NN: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

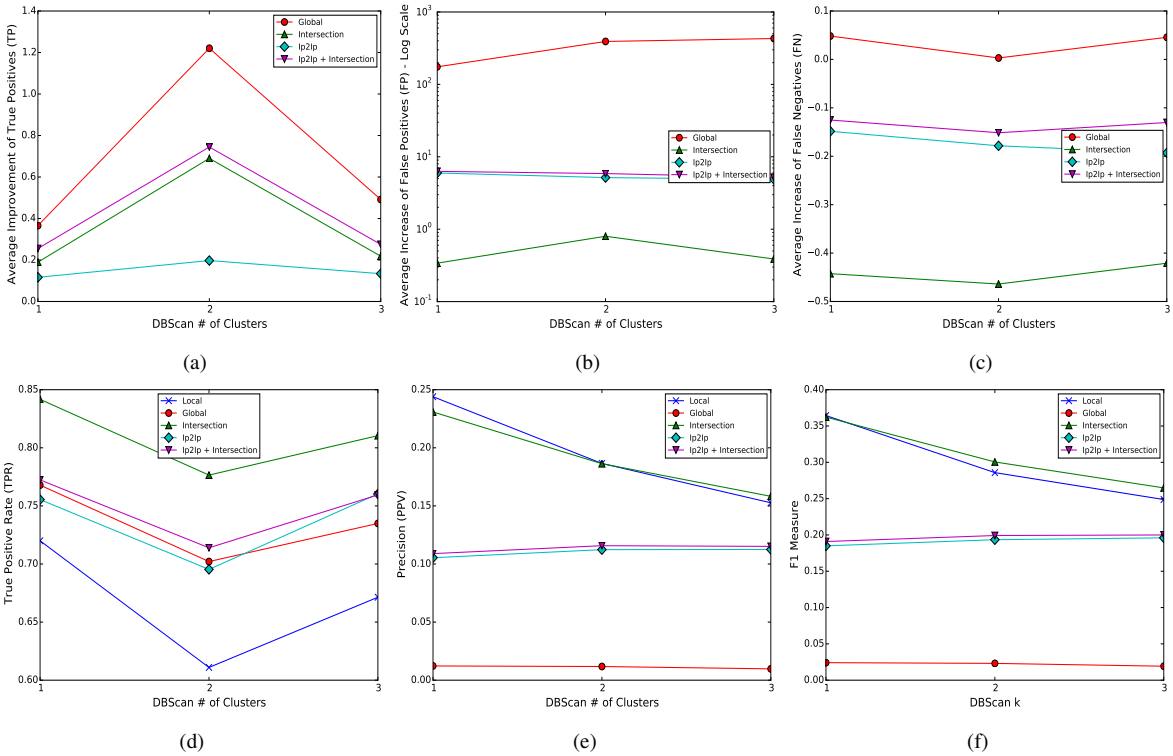


Figure 6: DBSCAN: (a) TP improvement, (b) FP increase (y-axis in log scale), (c) FN increase, (d) TPR, (e) Precision, (f) F1 measure.

clustering algorithms, this does not take the number of clusters as an input parameter, but relies on a threshold, eps, to define the maximum distance such that two samples are considered to be in the same neighborhood. For each time window, after tuning the parameter eps, the algorithm produces between 1 and 3 clusters. As the total number of collaborators range from 50 to 210 (out of the total 700), we conclude that the algorithm identifies a lot of outliers, i.e., organizations that are left out of the clusters, and results in the setting that involves the least collaborators.

Looking at Figure 6(a), we observe that the best TP_{impr} is

Setting		Max F1 [Sharing Intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglomerative	15	4.6	700	0.72	0.16	0.38 ± 3.51	0.71 ± 4.49	-0.42	0.27
k-means	5	5.8	280	0.73	0.19	0.44 ± 4.21	0.31 ± 1.47	-0.32	0.30
k-NN	15	6	240	0.74	0.19	0.27 ± 0.20	0.33 ± 0.28	-0.37	0.30
DBSCAN	1	33	33	0.84	0.23	0.18 ± 0.17	0.33 ± 0.22	-0.44	0.36

Table 2: Best Cases of our Experiments for F1.

Setting		Max TPR [Sharing Intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglomerative	1	70	700	0.76	0.15	0.50 ± 3.95	1.12 ± 6.98	-0.53	0.25
k-means	5	5.8	280	0.73	0.19	0.44 ± 4.21	0.31 ± 1.47	-0.32	0.30
k-NN	35	14	320	0.77	0.17	0.32 ± 0.21	0.51 ± 0.50	-0.49	0.28
DBSCAN	1	33	33	0.84	0.23	0.18 ± 0.17	0.33 ± 0.22	-0.44	0.36

Table 3: Best Cases of our Experiments for TPR.

Setting		Max PPV [Sharing Intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglomerative	25	2.8	700	0.69	0.16	0.33 ± 3.29	0.47 ± 2.23	-0.35	0.26
k-means	5	5.8	280	0.73	0.19	0.44 ± 4.21	0.31 ± 1.47	-0.32	0.30
k-NN	15	6	240	0.74	0.19	0.27 ± 0.20	0.33 ± 0.28	-0.37	0.30
DBSCAN	1	33	33	0.84	0.23	0.18 ± 0.17	0.33 ± 0.22	-0.44	0.36

Table 4: Best Cases of our Experiments for PPV.

Setting		Max TP Improvement [Sharing IP2IP+intersection]							
Clustering	k	Avg Size	#Coll.	TPR	PPV	TP_{impr}	FP_{incr}	FN_{incr}	$F1$
Agglomerative	1	70	700	0.67	0.11	0.52 ± 3.95	5.33 ± 16.9	-0.08	0.19
k-means	1	28	270	0.64	0.11	0.61 ± 5.36	3.55 ± 7.17	-0.17	0.18
k-NN	35	14	320	0.71	0.14	0.38 ± 0.25	1.51 ± 1.02	-0.19	0.23
DBSCAN	2	21.1	210	0.71	0.11	0.74 ± 6.04	5.86 ± 13.9	-0.15	0.20

Table 5: Best Cases of our Experiments for TP_{impr}.

produced when the algorithm builds 2 clusters. In this case, IP2IP+intersection achieves, on average, up to 0.70 TP_{impr} but with a relatively high FP_{incr} (5.86). We also look at (but do not plot) the TP_{impr} for every organization and notice that, with 2 clusters, there is one organization that improves dramatically more, while this does not happen when the algorithm builds 1 or 3 clusters. This fact highlights how, in certain settings, the benefits of collaboration are not so well distributed. Moreover, in Figures 6(b) and 6(c), we observe that for all sharing methods, FP_{incr} and FN_{incr} are rather steady and independent of the cluster sizes. Intersection achieves the lowest FP_{incr} (up to 0.8) and the biggest decrease in FN (-0.45), which explains its overall good performance in terms of PPV and TPR respectively. More precisely, Figure 6(d) shows how the algorithm behaves, in terms of TPR, for different number of clusters, with better results when building 1 cluster of 33 contributors or 3 clusters of 16 contributors on average, rather than 2 clusters of 22 collaborators as in TP_{impr} (up to 0.74 for IP2IP+intersection). As for precision (Figure 6(e)), intersection has a similar behavior to local (0.23), as with the other clustering algorithms, with worse PPV for IP2IP (0.11). Finally, Figure 6(f) plots the F1 measure, with intersection achieving the best result of 0.36 when one cluster is built. The F1 measures for IP2IP, and its combination with intersection, are both steady around 0.2.

5.3 Discussion

We now analyze the results of our experimental analysis and highlight a few interesting observations.

Summary of Results. We summarize the best results for each clustering algorithm, in terms of F1, recall, precision, and TP_{impr} respectively, in Tables 2–5. Note that intersection is that sharing mechanism that maximizes all metrics, except for TP_{impr}, which is instead maximized with IP2IP+intersection.

We observe that DBSCAN always yields the best performance, in terms of F1 (0.36), however, with one single cluster of 33 contributors, thus, only a relatively small fraction of organizations (4%) benefit from collaboration. Whereas, agglomerative clustering involves all 700 contributors and achieves $F1 = 0.27$. Both k-means and k-NN yield 0.30 in F1 including, respectively, 280 and 240 collaborators. Similarly, DBSCAN with one cluster yields the best results for TPR (0.84) and PPV (0.23) as shown in Tables 3–4. However, other collaboration settings include many more organizations with comparable results, e.g., k-NN with $k = 35$ and 320 collaborators achieves only 7% lower TPR (0.77), while k-means with $k = 15$ and 280 collaborators yields a 0.19 in PPV. In terms of TP_{impr}, DBSCAN reaches a maximum of 0.74 when it builds 2 clusters of size 21.1 on average, selecting 210 collaborators overall. Once again, slightly lower improvements are achieved with other clustering algorithms, but with more collaborators benefiting from sharing, as well as fewer FP.

Data sharing always helps organizations forecast attacks, compared to performing predictions locally. Predicting based on all data from collaborators yields the highest improvement in TP_{impr} – especially for bigger clusters – but with a dramatic increase in FP_{incr}. When organizations share correlated attacks (IP2IP), we observe a steady TP_{impr}, while sharing common attacks (intersection) outperforms the former when bigger clusters are formed. However, intersection introduces lower FP_{incr}, ultimately leading to better precision and F1 measures. IP2IP+intersection always outperforms the two separate methods in terms of TP_{impr}, thus, it is the recommended strategy if one only wants to maximize the number of predicted attacks – see Table 5.

Impact of cluster size. With agglomerative clustering, each organization is assigned to exactly one cluster and thus participates in/benefits from collaboration. We observe higher TPR for bigger clusters and, generally, a stable improvement in TP is achieved on average. Similar results are obtained with k-means when all organizations are assigned to clusters. However, when we set a distance threshold, creating more consistent clusters, we observe fluctuations in TPR: as clusters get smaller much faster (in relation to k value), IP2IP starts outperforming intersection. This indicates that correlated attacks can improve knowledge of organizations and enhance their local predictions, especially in smaller clusters.

With k-NN, a different behavior is observed: for smaller clusters, IP2IP achieves higher TPR (up to 0.7 for $k = 5$) but, as clusters get bigger, intersection yields the best results (up to 0.77 for $k = 35$). Then, as DBSCAN only builds large clusters, we observe fluctuations in TPR: for instance, intersection achieves 0.84 with one cluster, decreases to 0.77 with two clusters, and it increases again to 0.81 with three clusters. However, this fluctuation is not reflected by the TP_{impr} – in fact, with two clusters, we actually obtain the best improvement (0.74 for IP2IP+intersection), likely due to the way organizations are split into clusters, rather than FN_{incr} (which is steady and independent as shown in Figure 6(c)).

Overall, collaborating in big clusters leads to high TP_{impr} but at the same time it introduces significant FP_{incr}.

Increase/Improvement in TP/FP/FN. Table 5 shows that for all clustering algorithms, maximizing TP_{impr} always leads to higher FP_{incr}, from 1.51 of k-NN up to 5.86 of DBSCAN. The settings that maximize the F1 measure, TPR, and PPV, (when sharing intersection) also minimize FN_{incr}, e.g. agglomerative with $k = 1$ achieves -0.53 FN_{incr}. We also report the standard deviation of TP_{impr} and FP_{incr}, which provide an indication of the differences, in terms of "symmetry", of the benefits of collaboration—i.e., higher standard deviation shows that some organizations improve TP or increase FP much more than others. In general, we observe that (privacy-friendly) collaboration does yield a remarkable increase in TP but also in FP, which results in a limited improvement in accuracy (F1 measure) compared to predicting using local logs only. However, as discussed earlier, note that we count FP in a conservative way and that our main goal is really to measure the effect of different collaboration strategies on the prediction (as well

as comparing to Soldo et al. [37]), seeking to improve TP while keeping the increase in FP as low as possible.

Comparison with [37]. When comparing to Soldo et al. [37] (cf. Section 4), we observe that they achieve a higher maximum TP_{impr} (0.74 with DBSCAN, $k = 2$ vs 0.95), however, our privacy-preserving techniques outperform [37] in terms of recall (TPR) (e.g., with DBSCAN we reach 0.84 and with k-means 0.73, compared to their 0.66, i.e. up to 18% increase) as well as precision (0.23 with DBSCAN, $k = 1$ vs 0.08, i.e. up to 15% increase) and F1 measure (0.36 with DBSCAN, $k = 1$ vs 0.14). Note that, even if we consider settings involving more collaborators compared to DBSCAN, we still obtain appreciably better scores – e.g. agglomerative with $k = 15$ and 700 collaborators achieves $F1 = 0.27$, and k-means with 280 collaborators 0.30.

6. IMPLEMENTING AT SCALE

As discussed above, our system involves four steps: (1) secure computation of pairwise similarity, (2) clustering, (3) secure data sharing within the clusters, and (4) time-series prediction. To assess its scalability, we need to evaluate computation and communication complexities incurred by each step. Naturally, (1) and (3) dominate complexities as they require running a number of cryptographic protocols (involving public-key crypto) that depends on the number of organizations involved. In fact, clustering incurs a negligible overhead: on commodity hardware, to perform clustering with 1,000 organizations, it takes $6.1ms$ for k-means, $81ms$ for agglomerative and $5.2ms$ for k-NN ($k = 2$), and $6.3ms$ for DBSCAN ($eps = 0.2$). Also, time-series EWMA prediction requires $4.6\mu s$ per IP, so it takes $4.6ms$ for 1,000 IPs.

As we compute pairwise similarity based on the amount of common attacks between two organizations, and support its secure computation via PSI-CA [12], step (1) requires a number of protocol runs *quadratic* in the number of organizations. In our experiments (see details below), it takes $1.98s$ and 2.12MB bandwidth for one protocol execution, using 2048-bit moduli, with sets of size 4,000 (the average number of attacks observed by each organization). As for (3), i.e., secure within-cluster sharing of events related to common attacks (intersection), we rely on PSI-DT [13], and it takes $1.24s$ and 2.18MB for a single execution with the same settings. Therefore, complexities may quickly become prohibitive when more organizations are involved or more alerts are used.

Aiming to improve scalability, we introduce a variant supporting secure computation of pairwise similarity as well as secure log sharing *without* a quadratic number of public-key operations/quadratic communication overhead. Recall that we rely on a semi-trusted authority, STA, for clustering and coordination, which is assumed to follow protocol specifications and not to collude with other organizations, thus, we can actually use it to also help with secure computations. Inspired by Kamara et al.'s server-aided PSI [21] (see Appendix A), we extend our framework by replacing public-key cryptography operations with pseudo-random permutations (PRP), which we instantiate using AES. Specifically, we minimize interactions among pairs of organizations so that the complexity incurred by each of them is constant, while only imposing a minimal, linear communication overhead on STA.

Our extension involves four phases: (1) *setup*, where, as in [21], one organization generates a random key κ and sends it to the other organizations, (2) *encryption* (Algorithm 1), where each organization O_i evaluates the PRP on each entry d_j in their sets and encrypts the associated timestamp $time_j$, (3) *O2O computation* (Algorithm 2), where STA computes the magnitude of common attacks between each pair of organizations in order to perform clustering,

Algorithm 1 ENCRYPTION [All Organizations]

for each $O_i \in \mathcal{O}$ **do**
  $S_i \leftarrow \emptyset, E_i \leftarrow \emptyset, K_i \leftarrow \emptyset$
  for each $(d_j, time_j) \in D_i$ **do**
    for $cnt := 1$ **to** $COUNT(d_j)$ **do**
      $S_i \leftarrow S_i \cup \text{PRP}_\kappa(d_j||cnt)$
      $k_j \leftarrow H(d_j||cnt)$
      $E_i \leftarrow E_i \cup \text{Enc}_{k_j}(d_j, time_j)$
      $K_i \leftarrow K_i \cup k_j$
  Send S_i, E_i to STA and store K_i

Algorithm 2 O2O COMPUTATION [STA]

for each $O_i \in \mathcal{O}$ **do**
  for each $O_j \neq O_i$ **do**
    O2O$[i, j] \leftarrow |S_i \cap S_j|$
    Buff$[i, j] \leftarrow \{(\ell, E_{j_\ell}), \forall \ell \in \text{INDEX}(S_i \cap S_j)\}$
Perform Clustering on O2O$[\cdot, \cdot]$
Send relevant Buff$[\cdot, \cdot]$ entries to organizations in the same cluster

Algorithm 3 LOG SHARING [Organizations in C^*]

for each $O_i \in C^*$ **do**
  $S'_i \leftarrow \emptyset$
  for each $O_j \neq O_i \in C^*$ **do**
    for each $(\ell, E_{j_\ell}) \in$ Buff$[i, j]$ **do**
      $S'_i = S'_i \cup \text{Dec}_{k_\ell}(E_\ell)$

and (4) *log sharing* (Algorithm 3), where organizations in the same cluster C^* receive information about common attacks (S'_i-s).

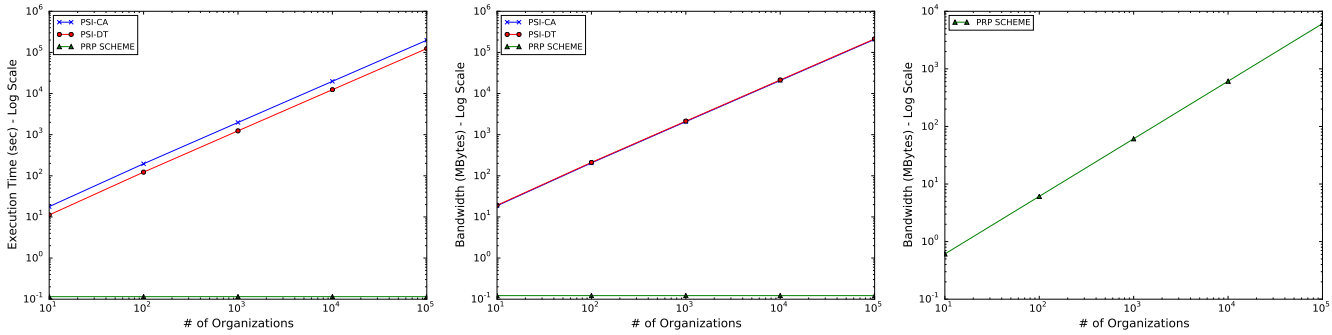
Note that building the O2O matrix is actually optimized using hash tables (i.e., *dense_hash_set* and *dense_hash_map* from Sparehash [19]). Also, since sets in our system are multi-sets, we concatenate counters to the IP address, so that the STA cannot tell which and how many IPs appear more than once.

A shortcoming of the PRP-based scheme is that if the STA colludes with an organization, it can decrypt data of all organizations. Whereas, using traditional PSI-CA/PSI-DT, collusion only leads to recovering a subset of datasets. Moreover, STA now also learns the size of each organization's dataset, although organizations could pad their sets by adding random entries, up to a fixed size.

Experimental Evaluation. To fully grasp the scalability of the server-aided extension, and compare it to using "traditional" PSI-CA and PSI-DT, we report execution times for increasing number of participating organizations. We benchmark the performance of PSI-CA [13] and PSI-DT [13] using 2048-bit moduli, modifying the OpenSSL/GMP-based C implementation in [14], as well as the PRP-based scheme presented above and inspired by Kamara et al.'s work [21]. Experiments are run using two 2.3GHz Intel Core i5 CPUs with 8GB of RAM connected via a 100Mbps Ethernet link.

Figures 7(a) and 7(b) plot computation and communication complexities incurred by an individual organization vis-à-vis the total number of organizations involved in the system, while Figure 7(c) reports the communication overhead introduced on the STA-side for the PRP scheme. Observe that complexities for PSI-CA/PSI-DT protocols on each organization grow linearly in the number of organizations (hence, quadratic overall). For instance, if 1,000 organizations are involved, it would take about 16 minutes per organization, each transmitting 1GB. Whereas, the PRP-based scheme incurs constant complexities on each organization ($57.6ms$ and 120KB) and an appreciably low communication overhead on the STA (about 100MB) for 1,000 organizations.

IP2IP. We also evaluate the IP2IP method whereby organizations



(a) (b) (c)

Figure 7: Computation (a) and communication (b) overhead at each organization for PSI-CA, PSI-DT, and PRP-based scheme, and communication overhead at the STA in PRP scheme (c).

interact with STA in order to discover cluster-wide correlated attacks. Assuming clusters of 100 organizations and an IP2IP matrix of $(2^{24} \cdot 2^{24})/2$ (recall from Section 5.2 that we consider the whole /24 IP space), we measure a $2.7s$ running time per organization with 41KB of bandwidth as well as a $0.07s$ overhead on the STA with 4.1MB bandwidth. Recall that we use the private Count-Min sketch based implementation by Melis et al. [27], which results in the private aggregation of 10,336 elements. Note that, even if clusters are bigger than 100, as detailed in [27], one can still perform private aggregation on multiple subgroups (e.g., of size 100) without endangering organizations' privacy.

Security. Our system does not leak *any* information about the logs of each organization to the STA, with or without using the server-aided variant. Clustering is performed over similarity measures computed obliviously to STA, and so does within-cluster data sharing. Privacy-preserving computation occurs by using existing secure protocols such as PSI-CA/PSI-DT by De Cristofaro et al. [13, 12]), server-aided PSI by Kamara et al. [21], as well as private recommendation via succinct sketches by Melis et al. [27]. Therefore, we do not provide any additional proofs in the paper as the security of our techniques straightforwardly relies on that of these protocols.

7. CONCLUSION

This paper presented a scalable privacy-friendly system geared to build highly predictive blacklists. A semi-trusted authority clusters organizations based on the similarity of their attack logs, without receiving logs in the clear. Entities in the same cluster then share, again in a privacy-preserving way, relevant logs with each other, and build more accurate predictive blacklists. We present a comprehensive set of measurements as we experiment with prior work as well as with four different clustering algorithms and three privacy-preserving sharing strategies, using real-world alerts collected from DShield.org. Our results show that previously proposed (non privacy-preserving) centralized algorithms [37] introduce a large number of false positives and false negatives, thus resulting in poor accuracy (a fact which was overlooked in prior work), and that our privacy-friendly techniques markedly outperform them.

In future work, we will experiment with other prediction algorithms and evaluate similar strategies to different collaborative security problems, such as spam filtering and malware detection.

8. REFERENCES

[1] myNetWatchman. http://www.mynetwatchman.com/, 2006.

[2] Facebook ThreatExchange. https://threatexchange.fb.com, 2015.

[3] S. Ackerman. Privacy experts question Obama's plan for new agency to counter cyber threats – The Guardian. http://gu.com/p/45yvz, 2015.

[4] B. Applebaum, H. Ringberg, M. Freedman, M. Caesar, and J. Rexford. Collaborative, privacy-preserving data aggregation at scale. In *PETS*, 2010.

[5] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. In *USENIX Security*, 2010.

[6] CERT UK. Cyber-security Information Sharing Partnership (CiSP). https://www.cert.gov.uk/cisp/, 2015.

[7] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[8] Communications Security, Reliability and Interoperability Council. U.S. Anti-Bot Code of Conduct for Internet service providers: Barriers and Metrics Considerations. https://transition.fcc.gov/bureaus/pshs/advisory/csric3/CSRIC_III_WG7_Report_March_%202013.pdf, 2013.

[9] G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *Journal of Algorithms*, 2005.

[10] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, M. K. Reiter, et al. Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces. In *NDSS*, 2007.

[11] M. Dacier, V.-H. Pham, and O. Thonnard. The WOMBAT Attack Attribution method: Some Results. In *Information Systems Security*. 2009.

[12] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and Private Computation of Cardinality of Set Intersection and Union. In *CANS*, 2012.

[13] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security*, 2010.

[14] E. De Cristofaro and G. Tsudik. Experimenting with fast private set intersection. In *TRUST*, 2012.

[15] M. Felegyhazi, C. Kreibich, and V. Paxson. On the potential of proactive domain blacklisting. In *LEET*, 2015.

[16] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, 2004.

[17] J. Freudiger, E. De Cristofaro, and A. Brito. Controlled Data Sharing for Collaborative Predictive Blacklisting. In *DIMVA*,

2015.

[18] O. Goldreich. *Foundations of Cryptography*, chapter 7.2.2. Cambridge Univ Press, 2004.

[19] D. Hide. Sparehash. https://github.com/sparsehash/sparsehash, 2013.

[20] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, 2003.

[21] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. Scaling private set intersection to billion-element sets. In *Financial Cryptography and Data Security*. 2014.

[22] S. Katti, B. Krishnamurthy, and D. Katabi. Collaborating against common enemies. In *ACM IMC*, 2005.

[23] L. Kissner and D. Song. Privacy-Preserving Set Operations. In *CRYPTO*, 2005.

[24] K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly Aggregation for the Smart-grid. In *Privacy Enhancing Technologies*, 2011.

[25] K. Lakkaraju and A. Slagell. Evaluating the utility of anonymized network traces for intrusion detection. In *SecureComm*, 2008.

[26] Y. Liu, A. Sarabi, J. Zhang, P. Naghizadeh, M. Karir, M. Bailey, and M. Liu. Cloudy with a Chance of Breach: Forecasting Cyber Security Incidents. In *USENIX Security*, 2015.

[27] L. Melis, G. Danezis, and E. De Cristofaro. Efficient Private Statistics with Succinct Sketches. In *NDSS*, 2016.

[28] G. Meng, Y. Liu, J. Zhang, A. Pokluda, and R. Boutaba. Collaborative Security: A Survey and Taxonomy. http://www3.ntu.edu.sg/home/ZhangJ/paper/csur.pdf, 2015.

[29] G. Moura, A. Sperotto, R. Sadre, and A. Pras. Evaluating third-party Bad Neighborhood blacklists for Spam detection. In *Integrated Network Management*, 2013.

[30] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. Botgrep: Finding p2p bots with structured graph analysis. In *USENIX Security Symposium*, 2010.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2011.

[32] B. Pinkas, T. Schneider, and M. Zohner. Faster Private Set Intersection based on OT Extension. In *USENIX Security*, 2014.

[33] P. Porras and V. Shmatikov. Large-scale collection and sanitization of network security data: risks and challenges. In *NSPW*, 2006.

[34] Red Sky Alliance. http://redskyalliance.org/.

[35] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In *WWW*, 2001.

[36] M. Sirivianos, K. Kim, and X. Yang. Socialfilter: Introducing social trust to collaborative spam mitigation. In *INFOCOM*, 2011.

[37] F. Soldo, A. Le, and A. Markopoulou. Predictive blacklisting as an implicit recommendation system. In *INFOCOM*, 2010.

[38] Symantec. DeepSight Threat Management System. http://tms.symantec.com, 2006.

[39] The White House. Executive order promoting private sector cybersecurity information sharing. http://1.usa.gov/1vISfBO, 2015.

[40] J. Zhang, P. A. Porras, and J. Ullrich. Highly predictive blacklisting. In *USENIX Security Symposium*, 2008.

APPENDIX

A. CRYPTOGRAPHY BACKGROUND

Adversarial Model. We use standard security models for secure two-party computation and consider semi-honest adversaries. In the rest of this paper, the term *adversary* refers to insiders, i.e., protocol participants. Outside adversaries are not considered, since their actions can be mitigated via standard network security techniques. Following definitions in [18], protocols secure in the presence of *semi-honest adversaries* assume that parties faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about the other party's input. This model is formalized by considering an ideal implementation where a trusted third party (TTP) receives the inputs of both parties and outputs the result of the defined function. Security in the presence of semi-honest adversaries requires that, in the real implementation of the protocol (without a TTP), each party does not learn more information than in the ideal implementation. Finally, we assume that parties do not collude with each other to recover other participants inputs.

Private Set Intersection (PSI): a cryptographic protocol between two parties, server and client, on input, respectively, $S = \{s_1, \ldots, s_w\}$ and $C = \{c_1, \ldots, c_v\}$. At the end, the client learns $S \cap C$. There are several PSI instantiations, with different complexities and cryptographic assumptions, ranging from those based on Oblivious Polynomial Evaluation (OPE) [16], to linear-complexity protocols based Oblivious PseudoRandom Functions (OPRFs) [13], as well as optimized garbled circuits [32] leveraging Oblivious Transfer Extension [20]. Naturally, the PSI definition above implies that only one party (client) learns the set intersection, however, in the semi-honest model, PSI can trivially be turned to a "mutual PSI" [23] (i.e., both parties learn the intersection) by executing PSI twice with inverted roles.

Private Set Intersection Cardinality (PSI-CA): a cryptographic protocol between two parties, server and client, on input, respectively, $S = \{s_1, \ldots, s_w\}$ and $C = \{c_1, \ldots, c_v\}$. At the end, the client learns $|S \cap C|$. That is, PSI-CA is a more stringent version of PSI as the client only learns how many items are in intersection.

While it is possible to modify garbled circuits based PSI constructions to support PSI-CA [32], to the best of our knowledge, there is no available description of the corresponding circuit or ready-to-use implementation, therefore, we use the special purpose PSI-CA protocol from [12]. This protocol is secure in the Random Oracle Model under the One-More Diffie-Hellman assumption in the presence of semi-honest adversaries. It incurs communication and computational complexities linear in the size of the sets: parties need to exchange $O(v + w)$ group items, and compute $O(v + w)$ modular exponentiations with short exponents. Similar to PSI, in the semi-honest model, two executions of PSI-CA with inverted roles yield a mutual PSI-CA where both parties learn the cardinality of the set intersection.

PSI with Data Transfer (PSI-DT): a cryptographic protocol between server and client on input, respectively, $S' = \{(s_1, data_1) \ldots, (s_w, data_w)\}$ and $C = \{c_1, \ldots, c_v\}$. At the end, the client obtains $\{(s_i, data_i \mid \exists\ c_j \text{ s.t. } s_i = c_j\}$. In other words, the client not only learns which items are in the intersection, but also gets related data records.

Special purpose protocols for PSI-DT have been proposed [16, 13], but we do not know of any available garbled circuits based instantiation. Hence, we use the PSI-DT protocol described in [13], secure in the Random Oracle Model under the One-More RSA assumption in the presence of semi-honest adversaries. It incurs communication and computational complexities linear in the size of the sets: parties need to exchange $O(v + w)$ group items, and compute $O(w)$ RSA-CRT exponentiations and $O(v)$ modular multiplications if one picks a small RSA public exponent (e.g., 3 or 17).

Once again, in the semi-honest model, two executions of PSI-DT with inverted roles trivially yield a mutual PSI-DT where both parties learn the intersection.

Server-aided PSI [21]. In [21], Kamara et al. propose a server-aided PSI relying on a semi-honest server: during a setup phase, parties $\{P_1, \ldots, P_n\}$ jointly generate a secret key κ for a pseudorandom permutation (PRP). Each party P_i then randomly permutes their set S_i, by computing $\text{PRP}_\kappa(S_i)$, and sends the result to the server. This then computes the intersection of the labels $\text{PRP}_\kappa(S_i), i = 1, \ldots, n$ and returns it to all the parties. Finally, each P_i outputs the inverse of PRP() over the intersection of the labels. The protocol is secure in the presence of a semi-honest server and honest parties, or a honest server and any collusion of malicious parties, if the PRP is secure.

Efficient Private Recommendation via Succinct Sketches [27]. A privacy-friendly recommender system based on Item-KNN [35] has been introduced by Melis et al. [27]. Their construction involves a "tally" server (the BBC in their application example) and a set of users (visitors of BBC's broadcasting site iPlayer). The main goal of their system is to train the recommender system using only aggregate statistics. Specifically, they build a global matrix of co-views (i.e., pairs of programs watched by the same user) in a privacy-preserving way, by relying on (i) private data aggregation based on secret sharing (inspired by [24]), and (ii) Count-Min sketches [9] to reduce the computation/communication overhead from linear to logarithmic in the size of the matrix, trading off an upper-bounded error with increased efficiency.

If M denotes the number of items (e.g., the programs on iPlayer in their application, or the number of IP addresses in ours), the compact representation of the IP2IP through the Count-Min Sketch has size $O(\log(M * M/2))$. More precisely, given parameters (ϵ, δ), the Count-Min Sketch is a matrix of size $L = d \times w$ where $d = \lceil \ln(M * M)/(2 * \delta) \rceil$ and $w = \lceil e/\epsilon \rceil$. Melis et al. [27] set $\epsilon = \delta = 0.01$, yielding, e.g., $L = 4,896$ for $M = 1,000$, and $L = 10,336$ for $M = 2^{24}$.

The parameters (ϵ, δ) give an upper bounded error for the estimated counters $\hat{c_i}$ amounting to $\hat{c_i} \leq c_i + \epsilon \sum_j |c_j|$ with probability $1 - \delta$, where c_i is the true element. As demonstrated empirically by Melis et al. [27], the error ultimately introduces a negligible impact on the accuracy of the aggregation as well as the recommendation. Finally, the computational overhead introduced by the cryptographic operations for private aggregation, as demonstrated experimentally in [27], are in the order of seconds even with thousands of items.

B. CLUSTERING ALGORITHMS

Agglomerative Clustering. Hierarchical Clustering algorithms build nested clusters by merging or splitting them successively. The hierarchy is represented as a tree, with the root being the unique cluster that gathers all the samples, and the leaves the clusters with only one sample. Agglomerative clustering performs hierarchical clustering using a bottom-up approach: each observation starts in its own cluster, and clusters are successively merged together. Different linkage criteria determine the actual metric used to merge, e.g., average linkage minimizes the average of the distances between all observations of pairs of clusters, while complete linkage minimizes the maximum distance between the observations of pairs of clusters.

k-means. k-means clustering separates samples in groups of equal variance, minimizing inertia or within-cluster sum of squares. The k-means algorithm requires the number of clusters to be specified as it divides a set of N samples X into k disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster "centroids" and the algorithm chooses centroids that minimize $\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2)$.

The algorithm includes three steps: (1) choosing the initial centroids, often by choosing k samples from X; (2) assigning each sample to its nearest centroid; and (3) creating new centroids by taking the mean value of all samples assigned to each previous centroid. The algorithm loops between (2) and (3) until the difference between the old and the new centroids is below a threshold.

k-Nearest Neighbors (k-NN). k-NN is a simple machine learning algorithm that finds a predefined number of training samples closest in distance to a new sample. The number of samples can be a user-defined constant and the distance can be any metric measure: standard Euclidean distance is the most common choice. In Section 5, we employ unsupervised k-NN to identify, for each organization, its most similar ones.

DBSCAN. Density-based spatial clustering of applications with noise (DBSCAN), given a set of points in some space, groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low density regions. The central component to DBSCAN are *core samples*, i.e., samples in areas of high density. A cluster is a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples). There are two parameters of the algorithm, *min_samples* and *eps*, defining density. Higher *min_samples* or lower *eps* indicate higher density necessary to form a cluster. In other words, a core sample is defined such that there exist *min_samples* other samples within a distance of *eps*. A cluster is a set of core samples, and can be built by recursively taking a core sample, finding all of its neighbors that are core samples, and so on.

C. CLUSTER SIZES

Figure 8 reports the average size of the clusters as well as the total number of collaborators involved in them with each clustering algorithm and various k values, as discussed in Section 5.2. Agglomerative clustering assigns all organizations to clusters (700 collaborators in total) with the average cluster size decreasing as k increases. When setting a cluster distance threshold, k-means yields a linear increase in the total number of collaborators up to 480 ($k = 35$) and the average cluster size decreases faster than agglomerative, in relation to k. k-NN with strong clusters involves between 220 and 320 collaborators and the average cluster size grows linearly in k (cluster sizes range from 1 to 14). Finally, for DBSCAN, the total number of collaborators peaks at 210 when it builds two clusters of average size 21.1.

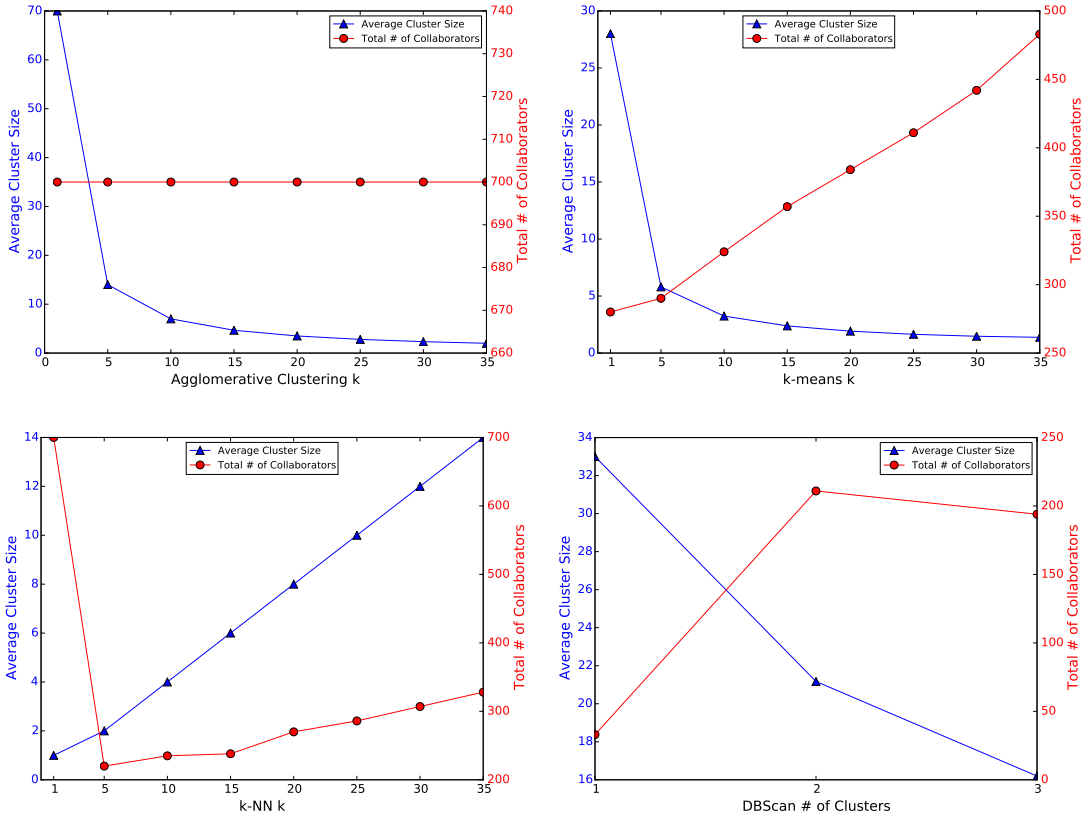


Figure 8: Average cluster sizes and total number of collaborators for: (a) Agglomerative Clustering, (b) k-means, (c) k-NN, and (d) DBScan.